

# CRC 알고리즘



- ◆ CRC 알고리즘을 이해한다.
- ◆ CRC 알고리즘을 이용한 데이터 무결성을 구현할 수 있다.

- 
- 1) 패리티 비트
  - 2) Check Sum
  - 3) CRC 알고리즘
  - 4) CRC TABLE 구현 및 확인
-

# Error Detection

---

## ◆ 정리

- 데이터 전송이 데이터값이 변경되는 경우가 발생
- 데이터가 올바르게 전송됐는지 확인하기 위한 값 필요
  - 체크섬값을 이용
- 체크섬값
  - 각각의 데이터값을 더한 값
- 전송값
  - `< data> <checksum>`
- Checksum값도 변경되는 경우가 발생할 수 있다.
- 데이터 값들이 변경되었는데도 checksum값과 같은 경우
  - Checksum 오류에도 보정 가능한 기능의 함수 필요

## 패리티 비트( parity bit )

---

```
char data = 'a';
```

```
if( hweight(data)%2 )  
    data |= 0x80;
```

```
01100001  
10000000 |  
-----  
11100001
```

```
char data = 'a';
```

```
if( hweight(data)%2 == 0 )  
    올바른 데이터  
else  
    변형된 데이터
```

```
11101001
```

## 패리티 비트( parity bit )

---

패리티 비트( parity bit ) : 한계점 -> 짝수개의 비트가 변형되면 error를 찾을 수 없다.

```
char data = 'a';
```

```
if( hweight(data)%2 )  
    data |= 0x80;
```

```
01100001  
10000000 |  
-----  
11100001
```

```
char data = 'a';
```

```
if( hweight(data)%2 == 0 )  
    올바른 데이터  
else  
    변형된 데이터
```

```
11101001
```

# Check Sum

struct iphdr {		송신부		
__u8	ihl:4,	45	4510	2DBFB
	version:4;	10	00c8	
__u8	tos;	00 c8	244c	DBFB
__be16	tot_len;	24 4c	4000	2 +
__be16	id;	40 00	4006	-----
__be16	frag_off;	40	0000	DBFD
__u8	ttl;	06	c0a8	
__u8	protocol;	00 00	3880	
__sum16	check;	c0 a8 38 80	c0a8	~ 1101 1011 1111 1101
__be32	saddr;	c0 a8 38 01	+ 3801	0010 0100 0000 0010
__be32	daddr;		-----	2 4 0 2
};			2DBFB	

# Check Sum

수신부

struct iphdr {				
__u8	ihl:4,	45	4510	2FFFD
	version:4;	10	00c8	
__u8	tos;	00 c8	244c	FFFD
__be16	tot_len;	24 4c	4000	2 +
__be16	id;	40 00	4006	-----
__be16	frag_off;	40	2402	FFFF
__u8	ttl;	06	c0a8	
__u8	protocol;	24 02	3880	
__sum16	check;	c0 a8 38 80	c0a8	~ 1111111111111111
__be32	saddr;	c0 a8 38 01	+ 3801	0000000000000000
__be32	daddr;		-----	0 0 0 0
};			2FFFD	

check sum의 한계점  
: 데이터중 하나는 더하기 되고 하나는 빼기가 되는 경우  
전체의 합에는 변동이 없으므로 error를 발견할 수 없다.

# CRC( Cyclic Redundancy Check )

송신부

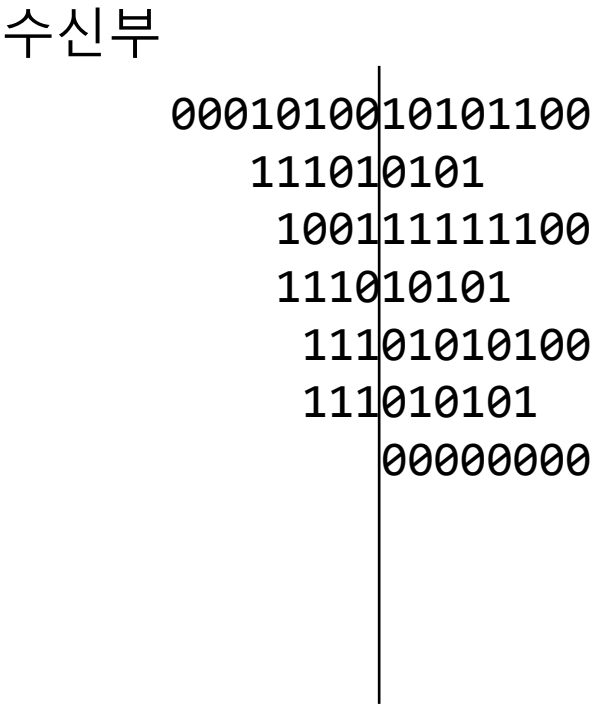
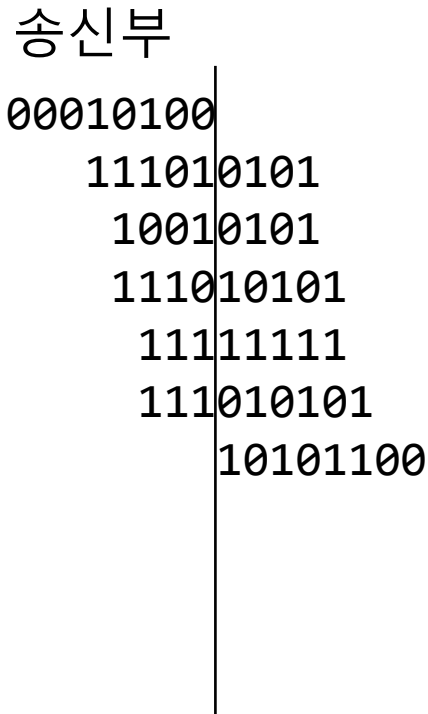
11010011101100	
1011	
1100011101100	
1011	
111011101100	
1011	
10111101100	
1011	
1101100	
1011	
110100	
1011	
11000	
1011	
1110	
1011	
101	0
101	1
	100

수신부

11010011101100	100
1011	
1100011101100	100
1011	
111011101100	100
1011	
10111101100	100
1011	
1101100	100
1011	
110100	100
1011	
11000	100
1011	
1110	100
1011	
101	100
101	100
	000

# CRC( Cyclic Redundancy Check )

```
data = 0x12;  
crc  = 0xac
```





# CRC( Cyclic Redundancy Check )

```
data[3] = {0x12,0x34,0x0};  
crc  = crc8(0x2d^0x34)  
data[2] = crc = 0xae;
```

